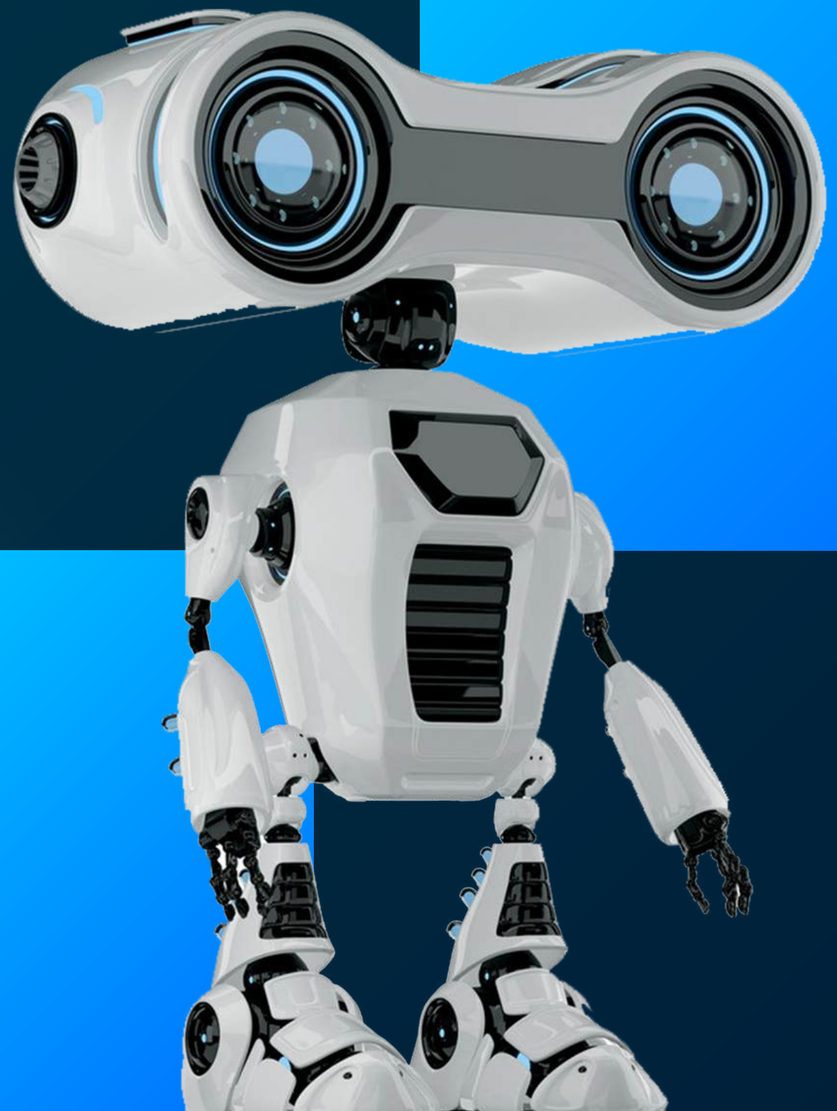


E - 3

COLLABORATION 3

Start Slide Presentations

목차

팀원 소개

팀장

남승록

전체 총괄 및 코드 개발 담당

팀원

이종화

코드 개발 어시스턴트 및 발표

팀원

권수빈

맵 제작 및 코드 분석

팀원

김인영

Main UI / DB 개발

OUR PROJECT

01 매설된 지뢰를 대신 처리해주는 로봇

02 드론과의 연계를 통해 효율적인 업무 수행

OUR VISION

01 드론을 통한 지뢰의 위치와 갯수를 탐지

02 GUI 화면으로 드론의 화면 송출 및 출동 좌표 전달

03 드론이 출동받은 좌표로 이동하여 임무 수행

04 귀환 명령에 따른 홈 위치의 복귀

04 지뢰를 찾은 좌표 및 시간을 DataBase에 저장

CODE REVIWE # DRONE

Drone을 gazebo에 띄우는 역할

sjtu_drone_description의
sjtu_drone_urdf.xacro 참조

sjtu_drone_gazebo.launch.py

```python
def generate_launch_description():
    use_sim_time = LaunchConfiguration("use_sim_time", default="true")
    use_gui = DeclareLaunchArgument("use_gui", default_value="true", choices=["true", "false"],
                                    description="Whether to execute gzclient")
    pkg_gazebo_ros = get_package_share_directory('gazebo_ros')
    xacro_file = os.path.join(
        get_package_share_directory("sjtu_drone_description"),
        "urdf", xacro_file_name
    )
    yaml_file_path = os.path.join(
        get_package_share_directory('sjtu_drone_bringup'),
        'config', 'drone.yaml'
    )

    robot_description_config = xacro.process_file(xacro_file, mappings={"params_path": yaml_file_path})
    robot_desc = robot_description_config.toxml()
    # get ns from yaml
    model_ns = "drone"
    with open(yaml_file_path, 'r') as f:
        yaml_dict = yaml.load(f, Loader=yaml.FullLoader)
        model_ns = yaml_dict["namespace"] #+ "/"
    print("namespace: ", model_ns)


    world_file_default = os.path.join(
        get_package_share_directory("sjtu_drone_description"),
        "worlds", "warehouse_mines_v0.0_w_color", "warehouse.world"
    )

    world_file = LaunchConfiguration('world', default=world_file_default)

    world = DeclareLaunchArgument(
        name='world',
        default_value=world_file_default,
        description='Full path to world file to load'
    )
```

(variable) pkg_gazebo_ros: str

CODE REVIWE # TURTLEBOT3

turtlebot3_world.launch.py

turtlebot3을 gazebo에 띄우는 역할

Model은 waffle로 설정

```python
def generate_launch_description():
    launch_file_dir = os.path.join(get_package_share_directory('turtlebot3_gazebo'), 'launch')
    # pkg_gazebo_ros = get_package_share_directory('gazebo_ros')  # 제거 가능

    use_sim_time = LaunchConfiguration('use_sim_time', default='true')
    x_pose = LaunchConfiguration('x_pose', default='-2.0')
    y_pose = LaunchConfiguration('y_pose', default='-0.5')

    world = os.path.join(
        get_package_share_directory('turtlebot3_gazebo'),
        'worlds',
        'turtlebot3_world.world'
    )
```

CODE REVIWE # COMBINED

combined.launch.py

Drone과 turtlebot을 gazebo에 띄우는 역할

sjtu_drone_bringup.launch.py와 turtlebot3_world.launch.py 참조

```
#######################################################################
# 3) turtlebot3_manipulation_bringup의 base.launch.py + Gazebo 로드
#######################################################################
manipulation_pkg = get_package_share_directory('turtlebot3_manipulation_bringup')

# base.launch.py
base_launch = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(manipulation_pkg, 'launch', 'base.launch.py')
    ),
    launch_arguments={
        'start_rviz': start_rviz,
        'prefix': prefix,
        'use_sim': use_sim,
    }.items(),
)

# gazebo_ros/launch/gazebo.launch.py
gazebo_launch = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(
            get_package_share_directory('gazebo_ros'),
            'launch',
            'gazebo.launch.py'
        )
    ),
    launch_arguments={
        'verbose': 'false',
        'world': world  # 커스텀 world 반영
    }.items(),
)

# 로봇팔+터틀봇3를 스폰 (spawn_entity.py)
spawn_manipulation_system = Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    arguments=[
        '-topic', 'robot_description',
        '-entity', 'turtlebot3_manipulation_system',
        '-x', x_pose,
        '-y', y_pose,
        '-z', z_pose,
        '-R', roll,
        '-P', pitch,
        '-Y', yaw,
    ],
    output='screen',
)
```

CODE REVIWE # MANIPULATION

turtlebot의 arm을 초기 위치로 고정시키는 역할

turtlebot_manipulation_bringup 패키지의 gazebo.launch.py 참조

combined.launch.py

```
#############################################################################
# 2) TurtleBot3 Manipulation 'gazebo.launch.py'에서 사용하던 인자들 정의
#############################################################################
start_rviz = LaunchConfiguration('start_rviz')
prefix = LaunchConfiguration('prefix')
use_sim = LaunchConfiguration('use_sim')

# 여기서 "원하는 world" 경로를 지정 (예: sjtu_drone_description 내 warehouse.world)
world = LaunchConfiguration(
    'world',
    default=PathJoinSubstitution([
        get_package_share_directory('sjtu_drone_description'),
        'worlds',
        'warehouse_mines_v0.0_w_color',
        'warehouse.world'
    ])
)

x_pose = LaunchConfiguration('x_pose', default='-2.00')
y_pose = LaunchConfiguration('y_pose', default='-0.50')
z_pose = LaunchConfiguration('z_pose', default='0.01')
roll  = LaunchConfiguration('roll',  default='0.00')
pitch = LaunchConfiguration('pitch', default='0.00')
yaw   = LaunchConfiguration('yaw',   default='0.00')

declare_args = [
    DeclareLaunchArgument('start_rviz', default_value='false',
                          description='Whether execute rviz2'),
    DeclareLaunchArgument('prefix', default_value='""',
                          description='Prefix of the joint and link names'),
    DeclareLaunchArgument('use_sim', default_value='true',
                          description='Start robot in Gazebo simulation.'),
    DeclareLaunchArgument('world', default_value=world,
                          description='Directory of gazebo world file'),
    DeclareLaunchArgument('x_pose', default_value=x_pose,
                          description='position of turtlebot3'),
    DeclareLaunchArgument('y_pose', default_value=y_pose,
                          description='position of turtlebot3'),
    DeclareLaunchArgument('z_pose', default_value=z_pose,
                          description='position of turtlebot3'),
    DeclareLaunchArgument('roll',   default_value=roll,
                          description='orientation of turtlebot3'),
    DeclareLaunchArgument('pitch',  default_value=pitch,
                          description='orientation of turtlebot3'),
    DeclareLaunchArgument('yaw',    default_value=yaw,
                          description='orientation of turtlebot3'),
]
```

CODE REVIWE # DRONE MOVE

Drone을 조종하는 역할

실제 지리의 위치까지 이동해서 확인하는 임무를 수행함

teleop.py

```
def __init__(self) -> None:
    super().__init__('teleop_node')
    # 퍼블리셔 토픽을 드론 플러그인 네임스페이스에 맞게 변경합니다.
    self.cmd_vel_publisher = self.create_publisher(Twist, '/simple_drone/cmd_vel', 10)
    self.takeoff_publisher = self.create_publisher(Empty, '/simple_drone/takeoff', 10)
    self.land_publisher = self.create_publisher(Empty, '/simple_drone/land', 10)

    # Velocity parameters
    self.linear_velocity = 0.0
    self.angular_velocity = 0.0
    self.linear_increment = 0.05
    self.angular_increment = 0.05
    self.max_linear_velocity = 1.0
    self.max_angular_velocity = 1.0

    # Start a timer to listen to keyboard inputs
    self.create_timer((1/30), self.read_keyboard_input)

def get_velocity_msg(self) -> str:
    return "Linear Velocity: " + str(self.linear_velocity) + "\nAngular Velocity: " + str(self.angular_velocity) + "\n"

def read_keyboard_input(self) -> None:
    """
    Read keyboard inputs and publish corresponding commands
    """
    while rclpy.ok():
        print(MSG + self.get_velocity_msg())
        key = self.get_key()
        if key.lower() == 'q':
            self.linear_velocity = min(self.linear_velocity + self.linear_increment, self.max_linear_velocity)
            self.angular_velocity = min(self.angular_velocity + self.angular_increment, self.max_angular_velocity)
        elif key.lower() == 'e':
            self.linear_velocity = max(self.linear_velocity - self.linear_increment, -self.max_linear_velocity)
            self.angular_velocity = max(self.angular_velocity - self.angular_increment, -self.max_angular_velocity)
        elif key.lower() == 'w':
            linear_vec = Vector3(x=self.linear_velocity)
            self.publish_cmd_vel(linear_vec)
        elif key.lower() == 's':
            self.publish_cmd_vel()
        elif key.lower() == 'x':
            linear_vec = Vector3(x=-self.linear_velocity)
            self.publish_cmd_vel(linear_vec)
        elif key == 'a':
            linear_vec = Vector3(y=self.linear_velocity)
            self.publish_cmd_vel(linear_vec)
```

CODE REVIWE # DRONE_POSE

Drone의 x, y, z 좌표를 Publish하는 역할

이후 GUI에서 /drone_position 토픽을 통해 좌표를 보내기 위해 사용

drone_pose_subscriber.py

```
    def __init__(self):
        super().__init__('drone_pose_subscriber')

        # 드론 오도메트리 토픽 구독
        self.subscription = self.create_subscription(
            Odometry,
            '/simple_drone/odom',  # 실제 드론 오도메트리 토픽 (플러그인/세팅에 따라 다를 수 있음)
            self.odom_callback,
            10
        )

        # 드론 좌표를 퍼블리시할 토픽 (Point 메시지)
        self.pose_publisher = self.create_publisher(Point, 'drone_position', 10)

        self.get_logger().info('DronePoseSubscriber node started!')

    def odom_callback(self, msg: Odometry):
        # 오도메트리에서 x,y,z 추출
        x = msg.pose.pose.position.x
        y = msg.pose.pose.position.y
        z = msg.pose.pose.position.z

        # Point 메시지 생성 후 퍼블리시
        point_msg = Point(x=x, y=y, z=z)
        self.pose_publisher.publish(point_msg)

def main(args=None):
    rclpy.init(args=args)
    node = DronePoseSubscriber()
    rclpy.spin(node)
    node.destroy_node()
```

CODE REVIWE # MAIN_GUI

main_gui.py / def startRemoval

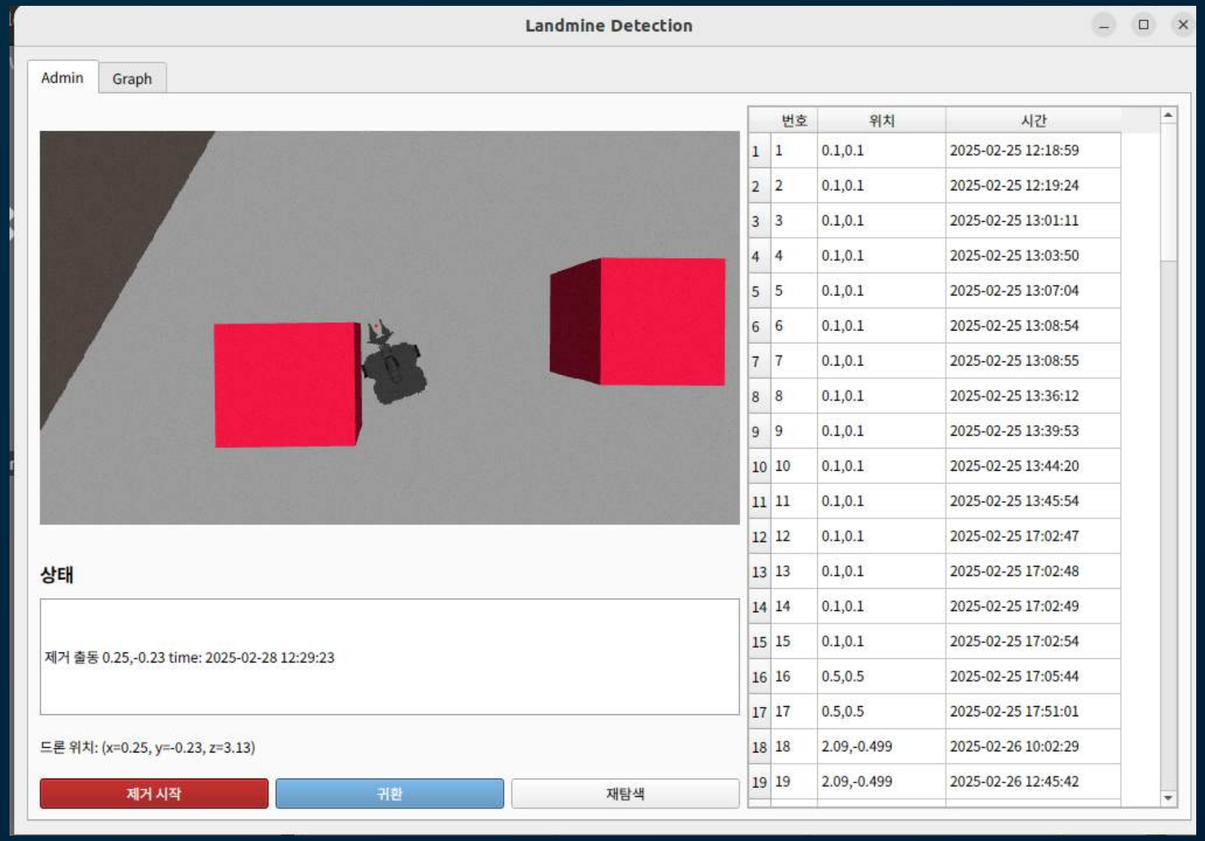
startRemoval 함수를 통해 turtlebot3에게 지뢰 제거 명령

전달받은 /drone_position의 x, y 좌표를 참조하여 turtlebot에게 전달

동시에 Database에 좌표, 시간 데이터를 저장

```python
def startRemoval(self):
    """
    제거 시작 버튼: 드론의 현재 (drone_x, drone_y) 위치로
    Turtlebot을 이동시킴(장애물 고려 없이 직선 이동)
    """
    current_time = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    # 1) DB 저장용으로 현재 드론 좌표를 문자열로 생성
    current_location = f"{self.drone_x:.2f},{self.drone_y:.2f}"
    self.saveToDatabase(current_location, current_time)

    # 2) 상태 업데이트
    message = f"제거 출동 {current_location} time: {current_time}"
    self.status_messages.append(str(message))
    self.updateStatusLabel()

    # 3) 목표 위치를 드론의 현재 x, y로 설정
    target_x = self.drone_x
    target_y = self.drone_y

    # 4) 직선 이동 함수 호출 (장애물 무시)
    self.move_straight_to_target(target_x, target_y)
```

CODE REVIWE # MAIN_GUI



	번호	위치	시간
1	1	0.1,0.1	2025-02-25 12:18:59
2	2	0.1,0.1	2025-02-25 12:19:24
3	3	0.1,0.1	2025-02-25 13:01:11
4	4	0.1,0.1	2025-02-25 13:03:50
5	5	0.1,0.1	2025-02-25 13:07:04
6	6	0.1,0.1	2025-02-25 13:08:54
7	7	0.1,0.1	2025-02-25 13:08:55
8	8	0.1,0.1	2025-02-25 13:36:12
9	9	0.1,0.1	2025-02-25 13:39:53
10	10	0.1,0.1	2025-02-25 13:44:20
11	11	0.1,0.1	2025-02-25 13:45:54
12	12	0.1,0.1	2025-02-25 17:02:47
13	13	0.1,0.1	2025-02-25 17:02:48
14	14	0.1,0.1	2025-02-25 17:02:49
15	15	0.1,0.1	2025-02-25 17:02:54
16	16	0.5,0.5	2025-02-25 17:05:44
17	17	0.5,0.5	2025-02-25 17:51:01
18	18	2.09,-0.499	2025-02-26 10:02:29
19	19	2.09,-0.499	2025-02-26 12:45:42

CODE REVIWE # MAIN_GUI

main_gui.py / def returnToBase

returnToBase 함수를 통해
turtlebot이 집으로 돌아가는 역할

초기 설정해놓은 위치로 좌표를 받아
이동함.

```python
def returnToBase(self):
    """
    Turtlebot을 (home_x, home_y) 위치로 귀환.
    (직선 이동 방식 그대로 사용 가능)
    """
    home_x = -0.968901
    home_y = 1.997310
    home_z = 0.0  # 지면 주행이므로 일반적으로 0.0

    message = f"Turtlebot 귀환 중: x={home_x}, y={home_y}"
    self.status_messages.append(message)
    self.updateStatusLabel()

    self.node.get_logger().info(f"Returning to base: ({home_x}, {home_y})")
    # 간단히 move_straight_to_target 이용
    self.move_straight_to_target(home_x, home_y)
```

CODE REVIWE # CAMERA

main_gui.py
/drone_bottom_camer

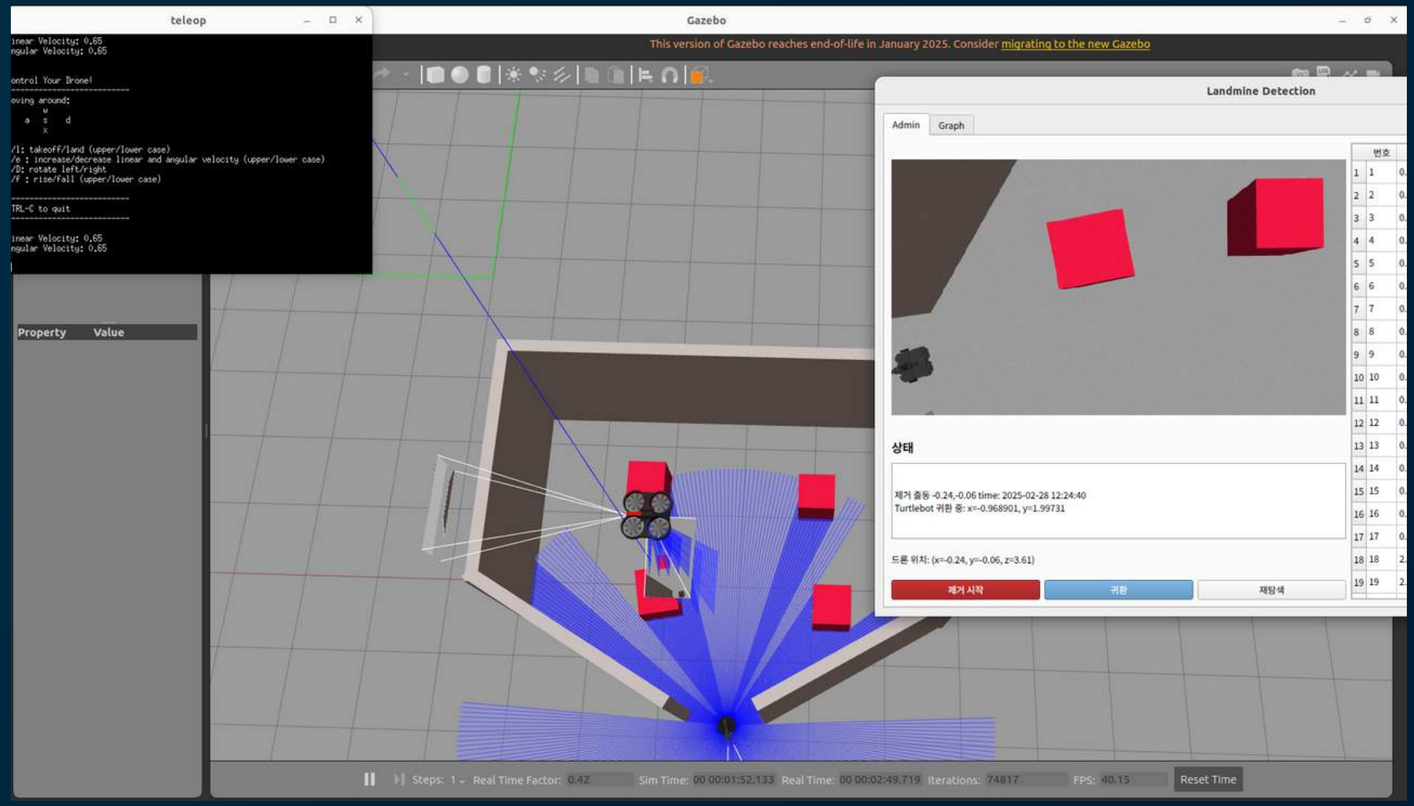
Drone의 bottom_camera를 GUI에 띄우는 작업

/simple_drone/bottom/image_raw를 통해 화면을 전달받아 opencv로 GUI에 이미지를 띄움

```python
# Drone 카메라 관련 subscription
self.bridge = CvBridge()
self.latest_image = None
self.node.create_subscription(Image, '/simple_drone/bottom/image_raw', self.image_callback, 10)
```

```python
def updateFrame(self):
    rclpy.spin_once(self.node, timeout_sec=0.001)
    if self.latest_image is not None:
        try:
            cv_image = self.bridge.imgmsg_to_cv2(self.latest_image, "bgr8")
        except CvBridgeError as e:
            print("CvBridge Error:", e)
            return
        cv_image = cv2.cvtColor(cv_image, cv2.COLOR_BGR2RGB)
        h, w, ch = cv_image.shape
        bytesPerLine = ch * w
        qImg = QImage(cv_image.data, w, h, bytesPerLine, QImage.Format_RGB888)
        qImg = qImg.scaled(600, 380, Qt.KeepAspectRatio)
        self.videoLabel.setPixmap(QPixmap.fromImage(qImg))
```

CODE REVIWE # CAMERA



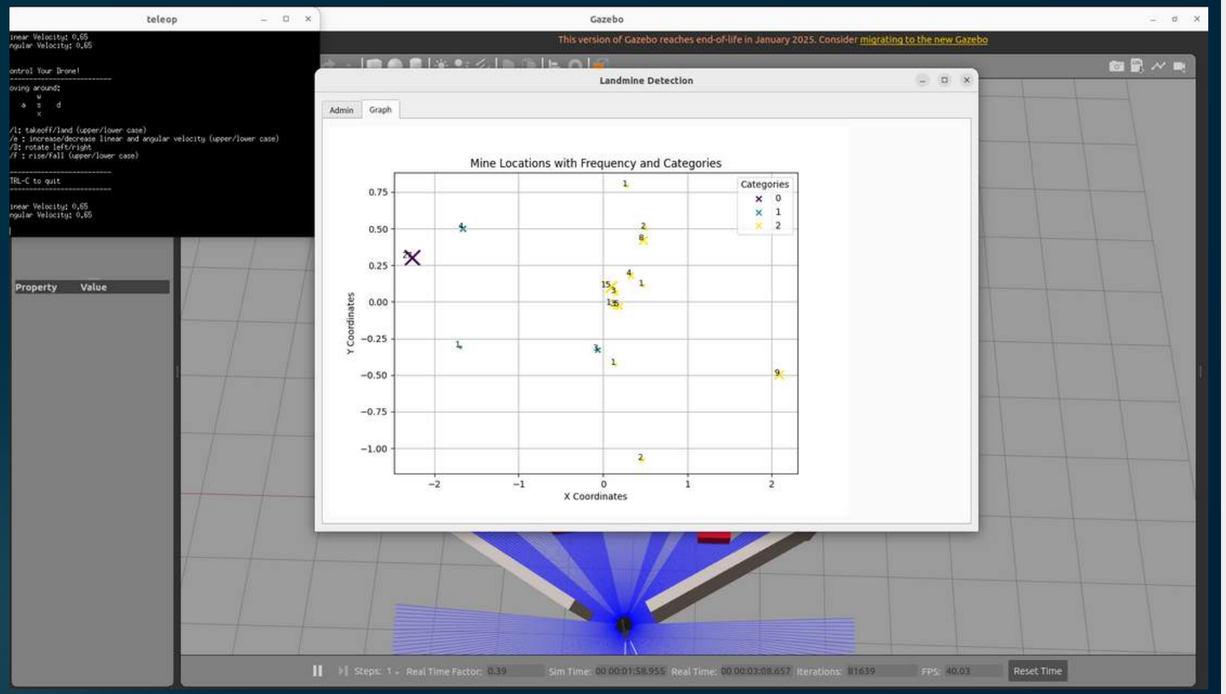
CODE REVIWE # CAMERA

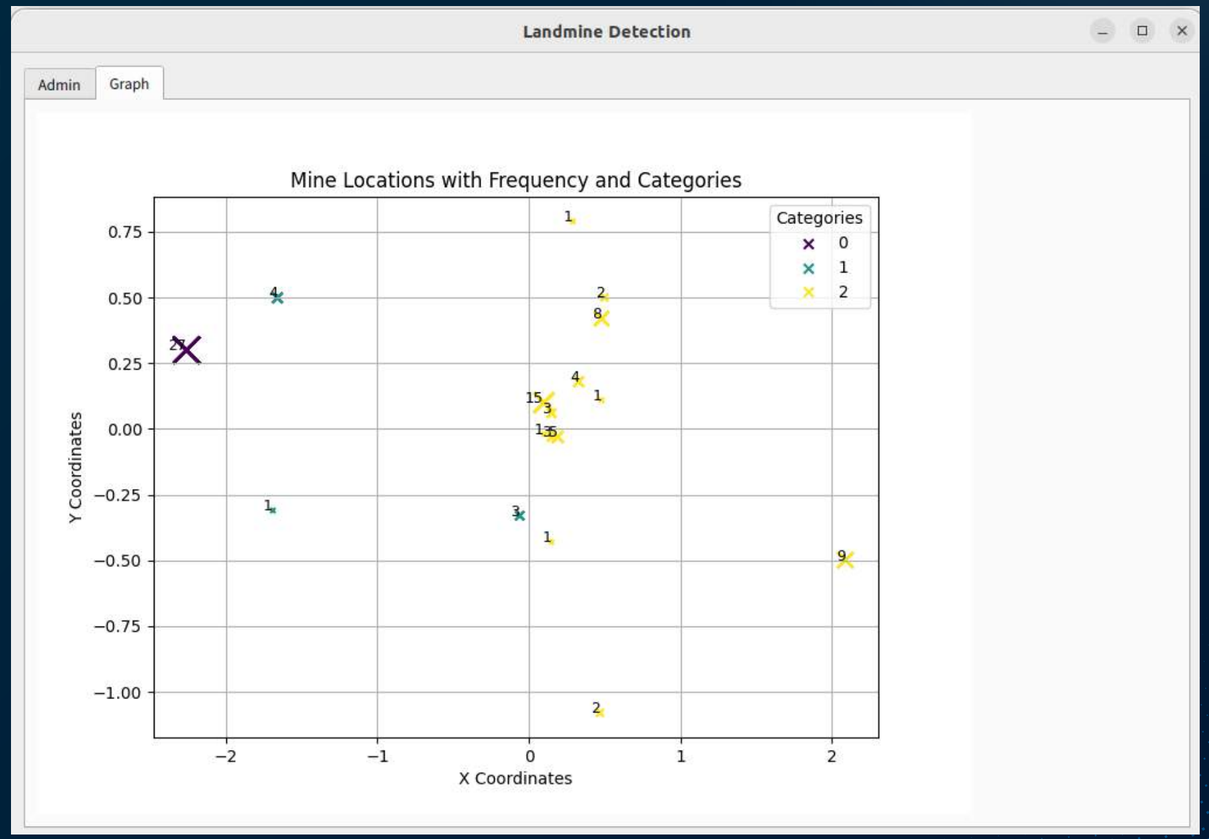
main_gui.py /updateGraph

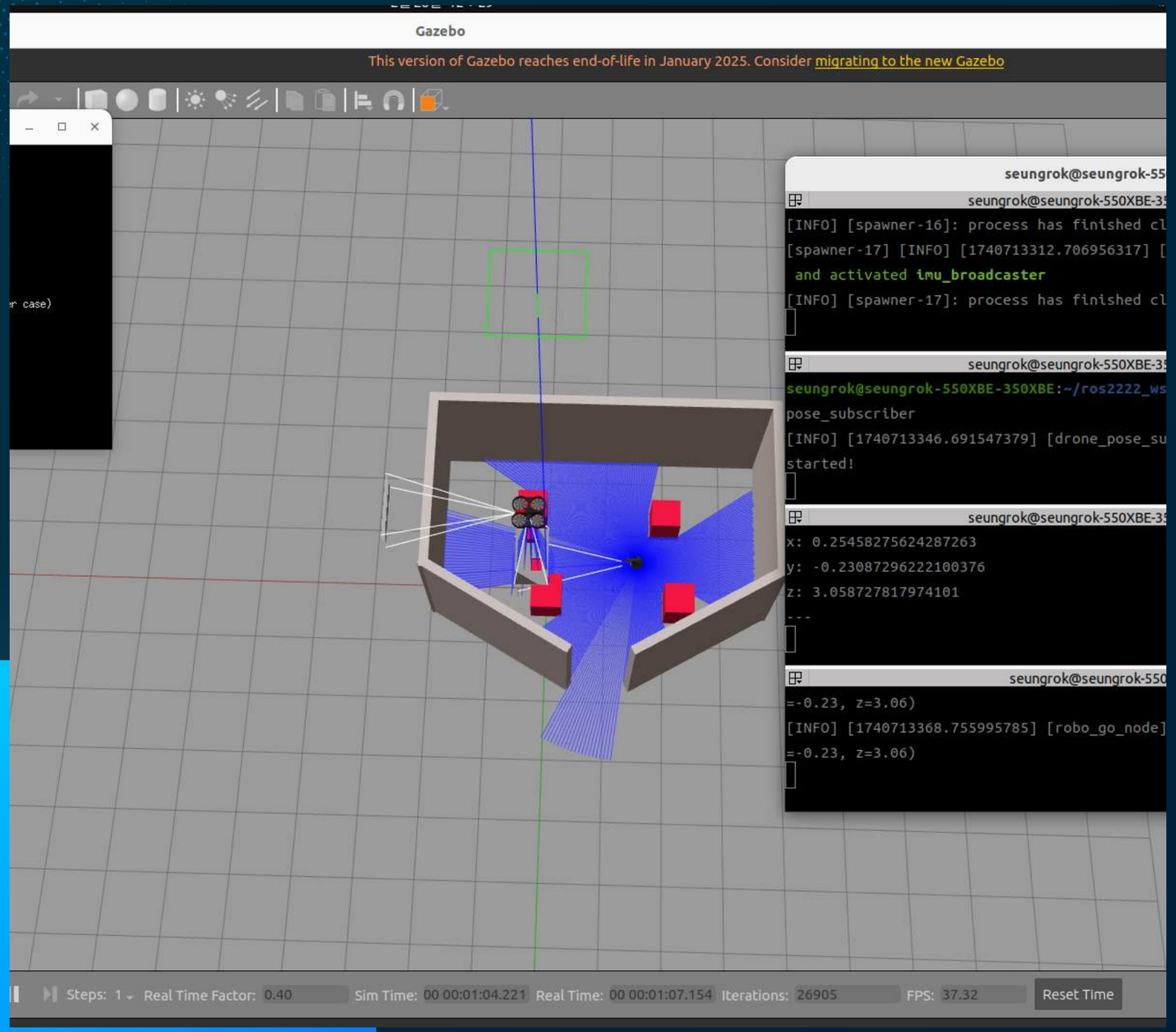
DataBase에 저장된 좌표의 위치를
그래프로 보여주는 역할

```python
def updateGraph(self):
    conn = sqlite3.connect('landmine.db')
    cursor = conn.cursor()

    cursor.execute("SELECT mine_location FROM search")
    rows = cursor.fetchall()

    mine_locations = []
    for row in rows:
        location_str = row[0].strip()
        match = re.match(r"^(-?\d+(\.\d+)?),\s*(-?\d+(\.\d+)?)$", location_str)
        if match:
            x = float(match.group(1))
            y = float(match.group(3))
            mine_locations.append((x, y))

    if mine_locations:
        x_coords, y_coords = zip(*mine_locations)
    else:
        x_coords, y_coords = [], []

    from collections import Counter
    mine_frequency = Counter(mine_locations)

    def categorize(x, y):
        if x < -2.0:
            return 'Close'
        elif -2.0 <= x <= 0:
            return 'Medium'
        else:
            return 'Far'

    categories = [categorize(x, y) for x, y in mine_locations]
    category_mapping = {'Close': 0, 'Medium': 1, 'Far': 2}
    category_numbers = [category_mapping[c] for c in categories]

    import matplotlib.pyplot as plt
    plt.figure(figsize=(8, 6))
    scatter = plt.scatter(
        x_coords, y_coords, c=category_numbers, cmap='viridis', marker='x',
        s=[mine_frequency[(x, y)]*10 for x, y in mine_locations]
    )

    plt.title("Mine Locations with Frequency and Categories")
    plt.xlabel("X Coordinates")
    plt.ylabel("Y Coordinates")
    plt.grid(True)
    plt.legend(*scatter.legend_elements(), title="Categories")

    for (x, y), freq in mine_frequency.items():
        plt.text(x, y, str(freq), fontsize=9, ha='right')

    pixmap = QPixmap("/tmp/mine_locations_with_frequency_and_categories.png")
    self.graphLabel.setPixmap(pixmap)
    plt.close()
```

CODE REVIWE # CAMERA







시연

collaboration 3

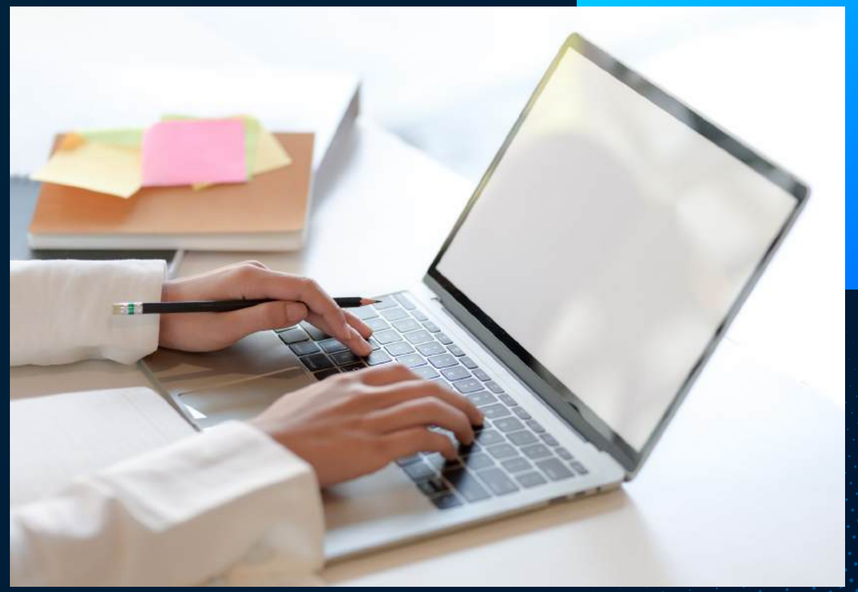
01 Time & Hardware Spac

02 Yolo training & Auto moving

01 Manipulation Setting

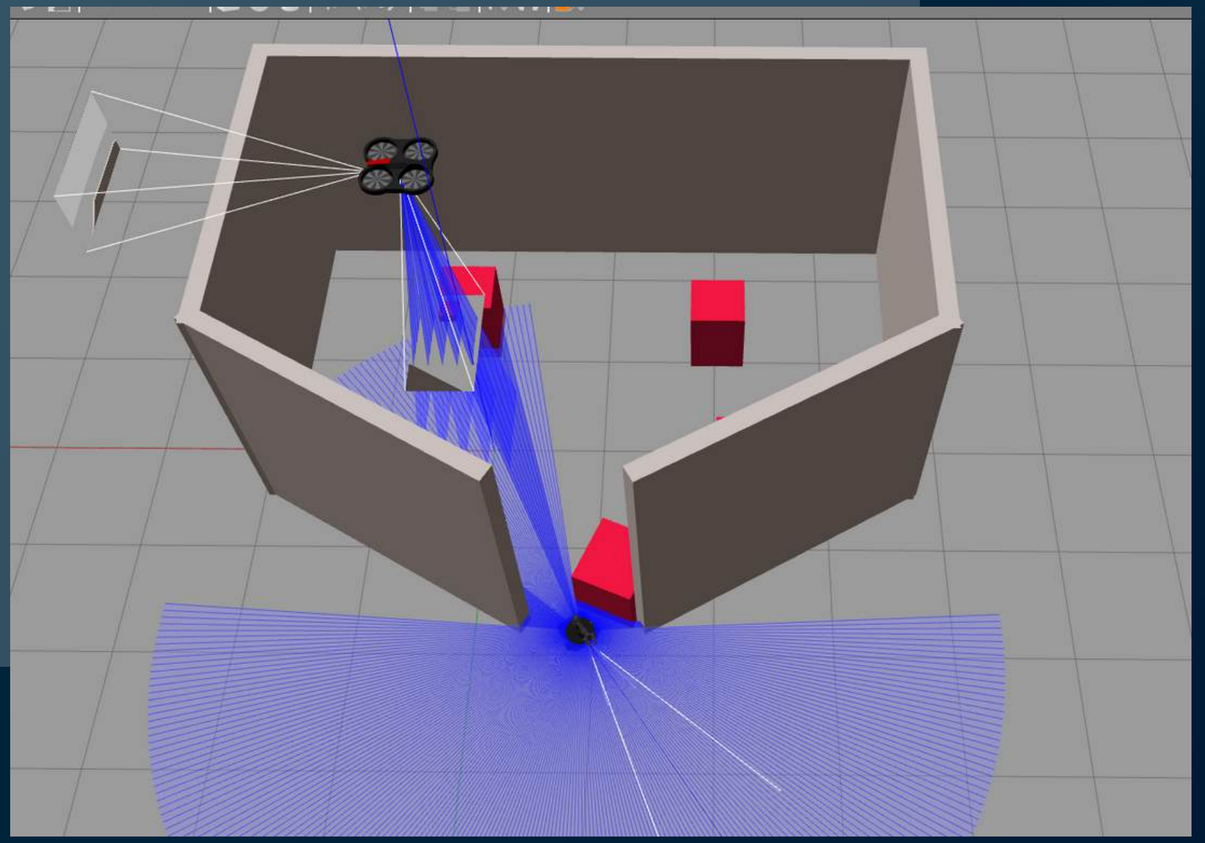
LIMITATIONS & SIGNIFICANCE

의미없는 자기 변명시간



Q&A

Questions n Answers



THANK YOU

감사합니다.